Integrated Simulation and Hardware-test Environment for Microcontroller Development

by
Andras Ferencz, Zsolt Kemecsei
Director, R&D engineer
Naturen Ltd.

Category:
Semiconductor

Products Used:
LabVIEW™
SQL-ToolKit
MS Visual Studio
PC-TIO-10
ATMIO16E10

The Challenge:
Creating a flexible, easy-to-use development and test environment for distributed intelligent controllers. Simulate plant in a PC environment, select and tune control algorithm for it. Generate documentation for target processor controller implementation. Test the real hardware in a simulated PC environment with its real inputs and outputs as far as possible.

The Solution:
Develop a LabVIEW Virtual Instrument library for simulating the functionality of the general 8 or 16-bit microcontrollers. Create Virtual Instrument library for connecting to Matlab/Simulink and Real-Time WorkShop, to be able to use models of the controlled system that were developed elsewhere. Create Virtual Front Panels of the controllers to speed up the design and test phase the of the user-friendly user interfaces for the production system. Create stimulating inputs for the production controllers and log their outputs using general, off-the-shelf PC cards from National Instruments. Develop a Virtual Instrument, which coordinates the use of different components and makes it easy to switch between Real and Simulated components any time during the development and test procedure.

Abstract
This paper demonstrates the strength of National Instruments hardware and software technology in a situation where it is important to build a heterogeneous environment with the need of simulation, prototype test, field measurements and production test. Since the same environment is used in different places in the R&D process, it speeds up the development and cuts back the costs. The ability to develop and tune control algorithms without being connected to the real controlled system or not using the prototype controller witch might not even exist at an early phase, opens a great opportunity for concurrent engineering especially in a Virtual Company. The fact that the simulation of the controlled system can be developed both in LabVIEW and Matlab environment helps to reduce the time of the specification for the customers since it is possible to use their model of the system that is required to be controlled.

Introduction
As a small private company mostly developing prototypes, individual process control, and measurement applications it is crucial for us to utilize the latest technology in the development process itself as well as in the applications. Our latest project is to develop customizable low cost microcontrollers for different applications. It is essential to develop general solutions to make it easy to adjust for more applications easily. Generally to design a process control application, it is necessary to have a good knowledge about the process. It does not only require expertise in several fields, but also requires the process or the controlled system for testing and parameter verification. That’s where simulation comes in. Since we are already developing integrated simulation and test systems for Electronic Control Units (ECU) in the automotive industry using LabVIEW, and National Instruments DAQ, the questions raised were trivial.

Is it possible to simulate the complete process including controller, and controlled system in a PC environment? Is it possible to use the same environment further more to stimulate the microcontroller prototype in real time to
verify the production code in it? This paper not only answers these questions, but shows the strengths of LabVIEW as a software integration environment in the development process.

**Simulation**

There are many ways to implement simulations in LabVIEW environment. Using native functions of G language, using simulation VIs or complete libraries, calling Matlab and Simulink from Matlab Script node and so on. However it is always an advantage to reuse already existing code and use it as an independent component as far as it is possible. That is why we choose to develop a VI library to be able to connect to standalone executable simulation models created with Matlab/Simulink and Real-Time WorkShop supplied by MathWorks. Since Matlab/Simulink is one of the leading products on the market for simulation and it is widely used in the industry it seemed obvious to have the possibility to use the knowledge already fed into those models and develop a controller around it, as if it would be the real system. This way, simulation and controller development could be separated and managed in different threads. While it seems to be easy to think about the controlled system as a black box separated from the controller there are many problems to solve:

- Getting information about the model
- Passing input values and receiving outputs
- Synchronizing the controller and the model

There are two ways to get information about the model structure, tunable parameters, and outputs. The first way is to use Matlab/Simulink built in commands to get these information via ActiveX server functionality of Matlab. We have developed a VI set to control Simulink models from LabVIEW witch still has an advantage on Matlab Script node. This advantage is the possibility to receive the complete “execute string” after the Matlab command. The native Matlab Script node can only handle numeric types. The “execute string” is important because most of the information you can get about the model hierarchy is string or cell type. This output then needs further phrasing and evaluation to extract the information and convert it to LabVIEW format. The following figure shows the main VI of the set we have developed to control Simulink from G.
However, the method shown above requires the complete model source. This may be difficult to share between organization considering classification and other parallel development issues. The other disadvantage of using Simulink directly is the performance. Especially with large sophisticated models it is running very slowly. This fact makes it impossible to use it for parameter tuning or controller optimization. Therefore we have developed a VI set to process the header file that Real-Time WorkShop creates when the stand-alone executable is created. Another text file is created with a simple script file that contains the initial values of tunable parameters. It was originally developed in another project and LabVIEW gave a great help reusing the VIs.

This is an easy to use tool to visualize the model hierarchy. It is easy to locate output signals, and select tunable parameters. Once we have the variable names both on input and output side we can use that list with the VI set for exchanging parameter values with the model.
TCP/IP
A stand-alone executable is generated by Real-Time WorkShop contains an external communication interface. It implements parameter download to the target, upload log data from the model and mechanism for triggering.

Figure 3 Matlab_Ext_Comm palette

The VI set shown above implements the connection on the LabVIEW side. It is based on a DLL, which contains the basic functionality, and the VIs are converting G data to the DLL and back.

Figure 4 A simple example of connecting to a Simulink Model
Simulation of the microcontroller
The next figure shows the main software loop of the controller simulation.

This plug-in architecture VI simulates the Turn-round Job System, implements a Watchdog and takes care of the synchronization. The plug-in Jobs are manipulating the global registers. A parallel loop is running representing the hardware of the microcontroller. This layer directs whether the register values are coming from converted real electrical signals through the DAQ system or from the simulation model. The same layer connects the registers to the real controller or virtual pin handled by the simulation.

Synchronization
Since it was the original design plan to create an environment where all the components handle their own execution time and the slowest controls the final simulation time, it couldn’t be implemented as a simple master-slave communication. Therefore as it is shown above, there are two parallel notification mechanisms. The first one notifies the controller that the model is ready. The input signals are available on the controller virtual pins. The second one tells the simulation loop that the controller is ready and all the output pins are available. When simulation is connected to the real DAQ system then the main loop in the hardware abstraction layer tells the simulation that all the measured inputs are available. When the simulation step is ready, the notifier tells to other Vis that the virtual pins are available and it is time to convert them to real electrical signals. While there are no speed requirements when all the components are simulated and it is only the user efficiency that is concerned, it is important to ensure that the simulation is running in Real-Time when it is connected to the real hardware.
Hardware connection diagram

- Command variables
- Controller parameters

**Simulated input signals:**
- Thermocouples
- RTDs
- 0.10 V
- 0..20 mA
- Discrete 24 VDC
- Discrete TTL/CMOS

**Output signals:**
- PWMs
- 0..10 V
- ±5 V
- 0..20 mA
- 4..20 mA
- Discrete 24 VDC
- Discrete TTL/CMOS

Figure 7 General hardware arrangement for the integrated environment

The general concept has been used in different ways.

Figure 8 Control software verification
It is used for parameter verification of the controlled system and parameter tuning of the controller. The same arrangement is used for production test of Power Unit and the controller. Measured data and parameters are stored in an ACCESS™ database.

**Conclusions**

With the help of the method and technology is described in this paper, we were able to create a development and test environment. The help of this flexible environment we have developed an industrial application, where we are able to control the temperature somewhere in a chamber, or in a solid part with only measuring 2 or more easily measurable temperature. The development including the environment development took less time than it would have taken with traditional methods.